INIT

login (server)		**login (client)**
	< -----	{msg: LOGIN, par: "nickname"}
(first player)		
(msg: LOGIN, ID: 0} **-> first login request**	----- >	
(already taken nickname)		
{msg: ERR, ErrorType: ALREADY_TAKEN_NICKNAME}		
(not first player)		
{msg: LOGIN}	----- >	

first login (server)		**first login (client)**
	< -----	{msg: NUM_PLAYERS, ID: int}

leader card choice (server)		**leader card choice (client)**
{msg: LEADER_CARD, ID: int, par1: leaderCard1, par2: leaderCard2, par3: leaderCard3, par4: leaderCard5}	----- >	
	< -----	{msg: LEADER_CARD, ID: int, par1: leaderCard1, par2: leaderCard2}
(on ok)		
{msg: OK, ID: int}	----- >	

new_player (server)		**new_player (client)**
{msg: NEW_PLAYER, ID: int, par: "nickname"}	----- >	

lost_player (server)		**lost_player (client)**
{msg: QUIT, ID: int}	----- >	

all_players (server)		**all_players (client)**
{msg: PLAYERS, ID: position}	----- >	

start game (server)		**start game (client)**
{msg: START_GAME, ID: int, par: num_giocatori}	----- >	
	< -----	{msg: TURN, ID: int}

market (server)		**market (client)**
{msg: MARKET, ID: int, par: Market}	----- >	

deckBoard (server		**deckBoard (client)**
{msg: DECKBOARD, ID: int, par: leaderCard[16]}	----- >	

TURN ACTION

buy card(server)		**buy card (client)**
	< -----	{msg: BUY_CARD, ID: int, par1: row, par2: column, par3: warehouse (0 or 1)}
(no available slots)		
{msg: ERR, ID: int, ErrorType: FULL_SLOT}	----- >	
(not enough resources)		
{msg: ERR, ID: int, ErrorType: NOT_ENOUGH_RESOURCES}	----- >	
(empty chosen deck)		
{msg: ERR, ID: int, ErrorType: EMPTY_DECK}	----- >	
(on ok)		
{msg: OK, ID: int}	----- >	
(more available slots)		
{msg: CHOSEN_SLOT, ID: int, par1: slot1, par2: slot2, par3: slot3} *(se solo 2 slot liberi, par3 = -1)*	----- >	
	< -----	{msg: CHOSEN_SLOT, ID: int, par: slot}
notifyAll {msg: BUY_CARD, ID: int, par1: card, par2: slot}	----- >	
notifyAll {msg: CARD_REMOVE, ID: int, par1: row, par2: column, par3: isEmpty (0 or 1), par4: new_card} *(se il mazzetto ora è vuoto, empty = 1 e new_card = -1)*	----- >	
notifyAll {msg: RESOURCE_AMOUNT, ID: int, par1: Resource, par2: amountWarehouse, par3: amountStrongbox} *(il messaggio viene inviato per ogni tipo di risorsa modificata, per un massimo di 4 messaggi)*	----- >	

take marble (server)		**take marble (client)**
	< -----	{msg: TAKE_MARBLE, ID: int, par1: row, par2: column}
{msg: TAKE_MARBLE, ID: int, par1: Marble1, par2: Marble2, par3: Marble3, par4: Marble4} *(se vengono inviate solo 3 biglie, Marble4 = null)*	----- >	
notifyAll(msg: MARKET_CHANGE, ID: int, par1: row, par2: column)	----- >	

use marble (server)		**use marble (client)**
	< -----	{msg: USE_MARBLE, ID: int, par: Marble}
(if WhiteMarble and 2 active WhiteConversionCard)		
{msg: WHITE_CONVERSION_CARD, ID: int, par1: leadrCard1, par2: leaderCard2}	----- >	
	< -----	{msg: WHITE_CONVERSION_CARD, ID: int, par1: leaderCard}
(on ok)		
{msg: OK, ID: int}	----- >	
(if red marble)		
notifyAll {msg: FAITH_POINTS_INCREASE, ID: int, par: totalFaithPoints}	----- >	
(if resource marble)		
notifyAll {msg: INCREASE_WAREHOUSE, ID: int, par1: Resource par2: depot (int)}	----- >	

(if marble is discarded)		
notifyAll {msg: FAITH_POINTS_INCREASE, ID: int, par: totalFaithPoints}	----- >	
(invia i punti fede degli altri giocatori incrementati di 1)		

switch (server)		**switch (client)**
	< -----	{msg: SWITCH_DEPOT, ID: int, par1: depot1, par2: depot2}
(switch not possible)	----- >	
{msg: ERR, ID: int, ErrorType: IMPOSSIBLE_SWITCH}		
(on ok)		
{msg: OK, ID: int}	----- >	
notifyAll {msg: SWITCH_DEPOT, ID: int, par1: depot1, par2: depot2}	----- >	

production power development card (server)		**production power development card (client)**
	< -----	{msg: DEVELOPMENT_CARD_POWER, ID: int, par1: slot, par2: warehouse (0 or 1)}
(not existing card)		
{msg: ERR, ID: int, ErrorType: EMPTY_SLOT}	----- >	
(not enough resources)		
{msg: ERR, ID: int, ErrorType: NOT_ENOUGH_RESOURCES"}	----- >	
(on ok)		
{msg: OK, ID: int}	----- >	
notifyAll {msg: END_PRODUCTION, ID: int}	----- >	

basic production power (server)		**basic production power (client)**
	< -----	{msg: BASIC_POWER, ID: int, par1: resourceDeleted1, par2: resourceDeleted2, par3: resourceObtained, par4: warehouse (0 or 1)}
(not enough resources)		
{msg: ERR, ID: int, ErrorType: NOT_ENOUGH_RESOURCES"}	----- >	
(on ok)		
{msg: OK, ID: int}	----- >	

additional production power (server)		**additional production power (client)**
	< -----	{msg: LEADER_CARD_POWER, ID: int, par1: leaderCard, par2: resourceObtained, par3: warehouse (0 or 1)}
(card is not active or is not an AddtionalProductionPowerCard)	----- >	
{msg: ERR, ID: int, ErrorType: WRONG_POWER}		
(on ok)		
{msg: OK, ID: int}	----- >	

end production power (server)		**end production power (client)**
	< -----	{msg: END_PRODUCTION, ID: int}
(no production power were casted)		
{msg: ERR, ID: int, ErrorType: ILLEGAL_OPERATION}	----- >	

(on ok)		
{msg: OK, ID: int}	----- >	
notifyAll {msg: FAITH_POINTS_INCREASE, ID: int, par: totalFaithPoints}	----- >	
notifyAll {msg: RESOURCE_AMOUNT, ID: int, par1: Resource, par2: amountWarehouse, par3: amountStrongbox}	----- >	
(il messaggio viene inviato per ogni tipo di risorsa modificata, per un massimo di 4 messaggi)		

leader card activation (server)		**leader card activation (client)**
	< -----	{msg: LEADER_CARD_ACTIVE, ID: int, par: leaderCard}
(card already active)		
{msg: ERR, ID: int, ErrorType: ALREADY_ACTIVE_LEADER_CARD}	----- >	
(card previously discarded)		
{msg: ERR, ID: int, ErrorType: ALREADY_DISCARD_LEADER_CARD}	----- >	
(not enough resources)		
{msg: ERR, ID: int, ErrorType: NOT_ENOUGH_RESOURCES	----- >	
(not enough cards)		
{msg: ERR, ID: int, ErrorType: NOT_ENOUGH_CARDS	----- >	
(on ok)		
{msg: OK, ID: int}	----- >	
notifyAll {msg: LEADER_CARD_ACTIVE, ID: int, par: leaderCard}		
(if extraDepot leader card)		
notifyAll {msg: EXTRA_DEPOT, ID: int, par: extraDepotResource}	----- >	

leader card discard (server)		**leader card discard (client)**
	< -----	{msg: LEADER_CARD_DISCARD, ID: int, par: leaderCard}
(card already active)		
{msg: ERR, ID: int, ErrorType: ALREADY_ACTIVE_LEADER_CARD}	----- >	
(card previously discarded)		
{msg: ERR, ID: int, ErrorType: ALREADY_DISCARD_LEADER_CARD}	----- >	
(on ok)		
{msg: OK, ID: int}	----- >	
notifyAll {msg: LEADER_CARD_DISCARD, ID: int, par: leaderCard}	----- >	
notifyAll {msg: FAITH_POINT_INCREASE, ID: int, par: amount}	----- >	

end_turn (server)		**end_turn (client)**
	< -----	{msg: END_TURN, ID: int}
notifyAll {msg: END_TURN, ID: int}	----- >	

CLIENT REQUEST

turn (server)		**turn (client)**
	< -----	{msg: TURN, ID: int}
{msg: TURN, ID: int, par: 0(notTurn) or 1(turn)}	----- >	

SERVER NOTIFY

ping (server)		**ping (client)**
{msg: PING, ID: int}	----- >	
	< -----	{msg: PING, ID: int}

vatican_report (server)		**vatican_report (client)**
{msg: VATICAN_REPORT, ID: int, par1: playerActivateVaticanReport, par2: totalVictoryPointsByVaticanReport}	----- >	

lost_player (server)		**lost_player (client)**
{msg: QUIT, ID: int}	----- >	

end_game (server)		**end_game (client)**
{msg: END: GAME, ID: winner, par1: winner_points, par2: winner_num_of_resources}	----- >	